Computer Architecture
Report #1: Register File

Joanna Whittam (15319524)

Feb. 24th 2017

1 VHDL components

1.1 Reg

1.1.1 Reg Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg is
        Port ( D: in STD_LOGIC_VECTOR (15 downto 0);
                load, CLK: in STD_LOGIC;
                Q: out STD_LOGIC_VECTOR (15 downto 0) );
end reg;

architecture Behavioral of reg is

begin
        process(CLK)
        begin
                if (rising_edge(CLK)) then
                        if load='1' then
                                Q <= D after 5 ns;
                        end if;
                end if;
        end process;
end Behavioral;
```

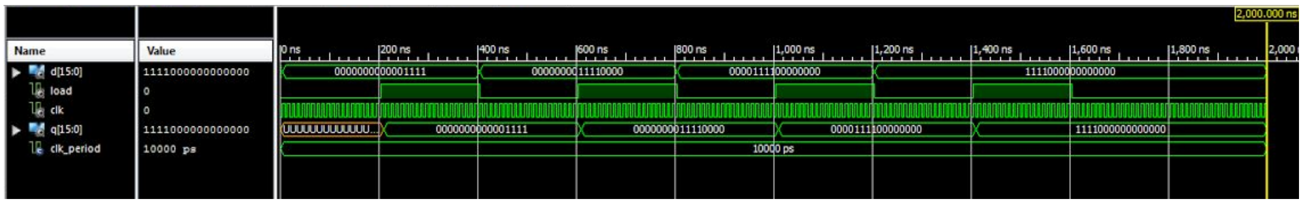
1.1.2 Reg Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY reg_tb IS
END reg_tb;

ARCHITECTURE behavior OF reg_tb IS

          -- Component Declaration for the Unit Under Test (UUT)

          COMPONENT reg
          PORT( D: IN  std_logic_vector(15 downto 0);
                     load, CLK: IN  std_logic;
                     Q: OUT  std_logic_vector(15 downto 0) );
```

```
         END COMPONENT;


        --Inputs
        signal D: std_logic_vector(15 downto 0) := (others => '0');
        signal load: std_logic := '0';
        signal CLK: std_logic := '0';

        --Outputs
        signal Q: std_logic_vector(15 downto 0);

        -- Clock period definitions
        constant CLK_period: time := 10 ns;

BEGIN
        -- Instantiate the Unit Under Test (UUT)
        uut: reg PORT MAP (
                  D => D,
                  load => load,
                  CLK => CLK,
                  Q => Q
        );

        -- Clock process definitions
        CLK_process :process
        begin
                    CLK <= '0';
                    wait for CLK_period/2;
                    CLK <= '1';
                    wait for CLK_period/2;
        end process;

       -- Stimulus process
        stim_proc: process
        begin
                    -- hold reset state for 50 ns.
                    wait for 5 ns;

                    D <= "0000000000001111";
                    wait for CLK_period*20;
                    load <= '1';
                    wait for CLK_period*20;
                    load <= '0';

                    D <= "0000000011110000";
                    wait for CLK_period*20;
                    load <= '1';
                    wait for CLK_period*20;
                    load <= '0';

                    D <= "0000111100000000";
                    wait for CLK_period*20;
                    load <= '1';
                    wait for CLK_period*20;
```

```
                        load <= '0';

                        D <= "1111000000000000";
                        wait for CLK_period*20;
                        load <= '1';
                        wait for CLK_period*20;
                        load <= ‘0';
                        wait;
                end process;
        END;
```

1.1.3 Reg Test Results



1.2 Mux2_16bit

In my file this entity is incorrectly named as mux2_4bit. This will be changed in later versions of the code.

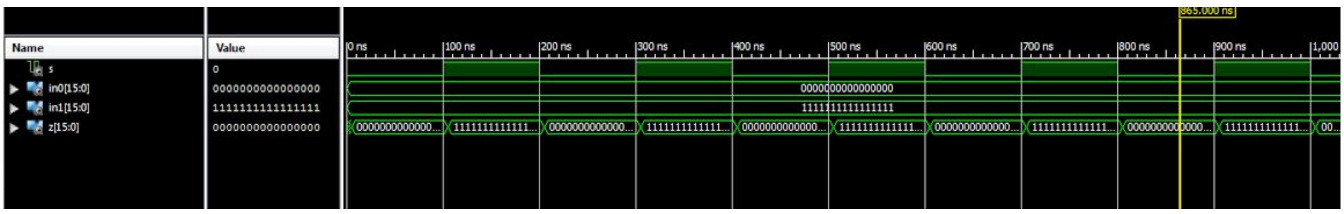
1.2.1 Mux2_16bit Code

```
        LIBRARY ieee;
        USE ieee.std_logic_1164.ALL;

        ENTITY mux2_4bit_tb IS
        END mux2_4bit_tb;

        ARCHITECTURE behavior OF mux2_4bit_tb IS

                -- Component Declaration for the Unit Under Test (UUT)
                COMPONENT mux2_4bit
                PORT(  S: in  STD_LOGIC;
                            In0, In1: in  STD_LOGIC_VECTOR (15 downto 0);
                            Z: out  STD_LOGIC_VECTOR (15 downto 0)
                );
                END COMPONENT;

                --Inputs
                signal S : std_logic := '0';
                signal In0 : std_logic_vector(15 downto 0) := (others => '0');
                signal In1 : std_logic_vector(15 downto 0) := (others => '0');

                --Outputs
                signal Z : std_logic_vector(15 downto 0);

        BEGIN
                -- Instantiate the Unit Under Test (UUT)
                uut: mux2_4bit PORT MAP (
                        S => S,
                        In0 => In0,
                        In1 => In1,
                        Z => Z
                );

                -- Stimulus process
                stim_proc: process
                begin
                        In0 <= "0000000000000000";
                        In1 <= "1111111111111111";

                        S <= '0';
                        wait for 100 ns;

                         S <= '1';
                        wait for 100 ns;
                end process;
        END;
```

1.2.2 Mux2_16bit Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux2_4bit_tb IS
END mux2_4bit_tb;

ARCHITECTURE behavior OF mux2_4bit_tb IS

        -- Component Declaration for the Unit Under Test (UUT)
        COMPONENT mux2_4bit
        PORT(     S: in  STD_LOGIC;
                In0, In1: in  STD_LOGIC_VECTOR (15 downto 0);
                Z: out  STD_LOGIC_VECTOR (15 downto 0)
        );
        END COMPONENT;

        --Inputs
        signal S : std_logic := '0';
        signal In0 : std_logic_vector(15 downto 0) := (others => '0');
        signal In1 : std_logic_vector(15 downto 0) := (others => '0');

        --Outputs
        signal Z : std_logic_vector(15 downto 0);

BEGIN
        -- Instantiate the Unit Under Test (UUT)
        uut: mux2_bit PORT MAP (
                S => S,
                In0 => In0,
                        In1 => In1,
                        Z => Z
            );

             -- Stimulus process
             stim_proc: process
             begin
                        In0 <= "0000000000000000";
                        In1 <= "1111111111111111";

                        S <= '0';
                        wait for 100 ns;

                        S <= '1';
                        wait for 100 ns;
        end process;
END;
```

1.2.3 Mux2_16bit Test Results



1.3 Mux8_16bit

1.3.1 Mux8_16bit Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity mux8_16bit is
        Port ( S0,S1,S2: in STD_LOGIC;
               In0,In1,In2,In3,In4,In5,In6,In7: in STD_LOGIC_VECTOR (15 downto 0);
               Z: out STD_LOGIC_VECTOR (15 downto 0));
end mux8_16bit;

architecture Behavioral of mux8_16bit is
begin
Z <=    In0 after 5 ns when S2='0' and S1='0' and S0='0' else
                In1 after 5 ns when S2='0' and S1='0' and S0='1' else
                In2 after 5 ns when S2='0' and S1='1' and S0='0' else
                In3 after 5 ns when S2='0' and S1='1' and S0='1' else
                In4 after 5 ns when S2='1' and S1='0' and S0='0' else
                In5 after 5 ns when S2='1' and S1='0' and S0='1' else
                In6 after 5 ns when S2='1' and S1='1' and S0='0' else
                In7 after 5 ns when S2='1' and S1='1' and S0='1';
end Behavioral;
```

1.3.2 Mux8_16bit Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux8_16bit_tb IS
END mux8_16bit_tb;

ARCHITECTURE behavior OF mux8_16bit_tb IS
        -- Component Declaration for the Unit Under Test (UUT)
        COMPONENT mux8_16bit
        PORT(  S0,S1,S2: in  STD_LOGIC;
                 In0,In1,In2,In3,In4,In5,In6,In7: in  STD_LOGIC_VECTOR (15 downto 0);
                 Z: out STD_LOGIC_VECTOR (15 downto 0));
        END COMPONENT;

        --Inputs
        signal S0 : std_logic := '0';
        signal S1 : std_logic := '0';
        signal S2 : std_logic := '0';
        signal In0 : std_logic_vector(15 downto 0) := (others => '0');
        signal In1 : std_logic_vector(15 downto 0) := (others => '0');
        signal In2 : std_logic_vector(15 downto 0) := (others => '0');
        signal In3 : std_logic_vector(15 downto 0) := (others => '0');
        signal In4 : std_logic_vector(15 downto 0) := (others => '0');
        signal In5 : std_logic_vector(15 downto 0) := (others => '0');
        signal In6 : std_logic_vector(15 downto 0) := (others => '0');
        signal In7 : std_logic_vector(15 downto 0) := (others => '0');
        signal Z : std_logic_vector(15 downto 0) := (others => '0');

BEGIN
        -- Instantiate the Unit Under Test (UUT)
        uut: mux8_16bit PORT MAP (
                S0 => S0,
                S1 => S1,
                S2 => S2,
                In0 => In0,
                In1 => In1,
                In2 => In2,
                In3 => In3,
                In4 => In4,
                In5 => In5,
                In6 => In6,
                In7 => In7,
                Z => Z
        );

        -- Stimulus process
        stim_proc: process
        begin
                In0 <= "0000000000000011";
                In1 <= "0000000000001100";
                In2 <= "0000000000110000";
                In3 <= "0000000011000000";
```

```
                    In4 <= "0000001100000000";
                    In5 <= "0000110000000000";
                    In6 <= "0011000000000000";
                    In7 <= "1100000000000000";
                     wait for 100 ns;

                    -- 0
                    S0<='0'; S1<='0'; S2<='0';
                    wait for 100 ns;

                    -- 1
                    S0<='1'; S1<='0'; S2<='0';
                    wait for 100 ns;

                    -- 2
                    S0<= '0'; S1<='1'; S2<='0';
                    wait for 100 ns;

                    -- 3
                    S0<= '1'; S1<='1'; S2<='0';
                    wait for 100 ns;
                    -- 4
                    S0<= '0'; S1<='0'; S2<='1';
                    wait for 100 ns;

                    -- 5
                    S0<= '1'; S1<='0'; S2<='1';
                    wait for 100 ns;

                    -- 6
                    S0<= '0'; S1<='1'; S2<='1';
                    wait for 100 ns;

                    -- 7
                    S0<='1'; S1<='1'; S2<='1';
                    wait;
            end process;
    END;
```

1.3.3 Mux8_16bit Test Results

1.4 Decoder

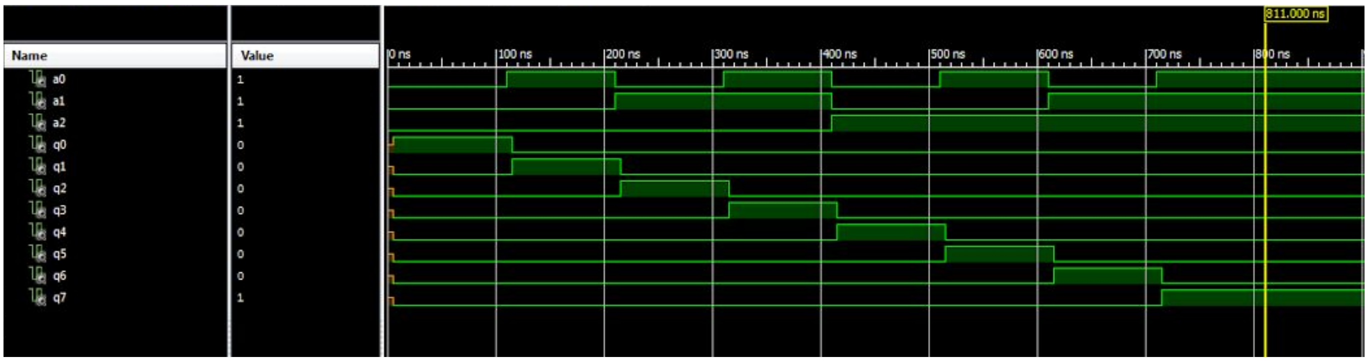
1.4.1 Decoder Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity decoder_3to8 is
        Port (    A0,A1,A2: in  STD_LOGIC;
                  Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7: out  STD_LOGIC );
end decoder_3to8;

architecture Behavioral of decoder_3to8 is
begin
        Q0 <=   ((not A2) and (not A1)      and (not A0))      after 5 ns;
        Q1 <= ((not A2) and (not A1)        and  A0)           after 5 ns;
        Q2 <= ((not A2) and  A1     and (not A0))      after 5 ns;
        Q3 <= ((not A2) and  A1     and  A0)           after 5 ns;
        Q4 <= ( A2          and (not A1)      and (not A0))      after 5 ns;
        Q5 <= ( A2          and (not A1)      and  A0)           after 5 ns;
        Q6 <= ( A2          and  A1           and (not A0))      after 5 ns;
        Q7 <= ( A2          and  A1           and  A0)           after 5 ns;
end Behavioral;
```

1.4.2 Decoder Test Bench

```
        LIBRARY ieee;
        USE ieee.std_logic_1164.ALL;

        ENTITY decoder_3to8_tb IS
        END decoder_3to8_tb;

        ARCHITECTURE behavior OF decoder_3to8_tb IS

                  -- Component Declaration for the Unit Under Test (UUT)
                  COMPONENT decoder_3to8
                  PORT(          A0,A1,A2: in  STD_LOGIC;
                                 Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7: out  STD_LOGIC );
                  END COMPONENT;

                 --Inputs
                 signal A0 : std_logic := '0';
                 signal A1 : std_logic := '0';
                 signal A2 : std_logic := '0';

                --Outputs
                signal Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7: std_logic;

BEGIN
                -- Instantiate the Unit Under Test (UUT)
                uut: decoder_3to8 PORT MAP (
                          A0 => A0,
			A1 => A1,
			A2 => A2,
			Q0 => Q0,
			Q1 => Q1,
			Q2 => Q2,
			Q3 => Q3,
			Q4 => Q4,
			Q5 => Q5,
			Q6 => Q6,
			Q7 => Q7
		);

		-- Stimulus process
		stim_proc: process
		begin
			-- hold reset state for 100 ns.
			wait for 10 ns;

			-- 0
			A0<='0'; A1<='0'; A2<='0';
			wait for 100 ns;

			-- 1
			A0<='1'; A1<='0'; A2<='0';
			wait for 100 ns;

			-- 2
			A0<='0'; A1<='1'; A2<='0';
			wait for 100 ns;

			-- 3
			A0<='1'; A1<='1'; A2<='0';
			wait for 100 ns;

			-- 4
			A0<='0'; A1<='0'; A2<='1';
			wait for 100 ns;

			-- 5
			A0<='1'; A1<='0'; A2<='1';
			wait for 100 ns;

			-- 6
			A0<='0'; A1<='1'; A2<='1';
			wait for 100 ns;

			-- 7
			A0<='1'; A1<='1'; A2<='1';
			wait;
		end process;
	END;
```

1.4.3 Decoder Test Results



2 Register File

2.1 Register_file Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity register_file is
        Port ( src_s0, src_s1, src_s2: in  STD_LOGIC;
                des_A0, des_A1, des_A2: in  STD_LOGIC;
                CLK: in  STD_LOGIC;
                data_src: in  STD_LOGIC;
                data: in  STD_LOGIC_VECTOR (15 downto 0);
                reg0, reg1, reg2, reg3: out  STD_LOGIC_VECTOR (15 downto 0)
                reg4, reg5, reg6, reg7: out  STD_LOGIC_VECTOR (15 downto 0) );
end register_file;

architecture Behavioral of register_file is

--COMPONENTS
        component decoder_3to8 is
        Port (  A0,A1,A2: in  STD_LOGIC;
                Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7: out  STD_LOGIC );
        end component;

        component mux2_4bit is
        Port (   S: in  STD_LOGIC;
                In0, In1: in  STD_LOGIC_VECTOR (15 downto 0);
                Z: out  STD_LOGIC_VECTOR (15 downto 0));
        end component;

        component mux8_16bit is
        Port (  S0,S1,S2: in STD_LOGIC;
                In0,In1,In2,In3,In4,In5,In6,In7: in STD_LOGIC_VECTOR (15 downto 0);
                Z: out STD_LOGIC_VECTOR (15 downto 0));
        end component;
        component reg
        Port (  D: in STD_LOGIC_VECTOR (15 downto 0);
                load, CLK: in STD_LOGIC;
                Q: out STD_LOGIC_VECTOR (15 downto 0) );
        end component;

--SIGNALS
        signal  load_r0, load_r1, load_r2, load_r3, load_r4, load_r5, load_r6, load_r7: std_logic;
        signal  reg0_q, reg1_q, reg2_q, reg3_q, reg4_q, reg5_q, reg6_q, reg7_q,
                data_src_mux_out, src_reg : std_logic_vector(15 downto 0);

begin
--PORT MAPS
        --Registers
        reg00: reg PORT MAP(
                D => data_src_mux_out,
                load => load_r0,
                Clk => Clk,
                Q => reg0_q );

        reg01: reg PORT MAP(
                D => data_src_mux_out,
                load => load_r1,
                Clk => Clk,
                Q => reg1_q );

        reg02: reg PORT MAP(
                D => data_src_mux_out,
                load => load_r2,
                Clk => Clk,
                Q => reg2_q );

        reg03: reg PORT MAP(
                D => data_src_mux_out,
                load => load_r3,
                Clk => Clk,
                Q => reg3_q );

        reg04: reg PORT MAP(
                D => data_src_mux_out,
                load => load_r4,
                Clk => Clk,
                Q => reg4_q );

        reg05: reg PORT MAP(
                D => data_src_mux_out,
                load => load_r5,
                Clk => Clk,
                Q => reg5_q );

        reg06: reg PORT MAP(
                D => data_src_mux_out,
                load => load_r6,
			Clk => Clk,
			Q => reg6_q );

		reg07: reg PORT MAP(
			D => data_src_mux_out,
			load => load_r7,
			Clk => Clk,
			Q => reg7_q );

		--Decoder
		decoder: decoder_3to8 PORT MAP(
			A0 => des_A0,
			A1 => des_A1,
			A2 => des_A2,
			Q0 => load_r0,
			Q1 => load_r1,
			Q2 => load_r2,
			Q3 => load_r3,
			Q4 => load_r4,
			Q5 => load_r5,
			Q6 => load_r6,
			Q7 => load_r7 );

		-- Multiplexers
		data_src_mux2_4bit: mux2_4bit PORT MAP(
			S => data_src,
			In0 => data,
			In1 => src_reg,
			Z => data_src_mux_out);

		Inst_mux8_16bit: mux8_16bit PORT MAP(
			S0 => src_s0,
			S1 => src_s1,
			S2 => src_s2,
			In0 => reg0_q,
			In1 => reg1_q,
			In2 => reg2_q,
			In3 => reg3_q,
			In4 => reg4_q,
			In5 => reg5_q,
			In6 => reg6_q,
			In7 => reg7_q,
			Z => src_reg);

		reg0 <= reg0_q;
		reg1 <= reg1_q;
		reg2 <= reg2_q;
		reg3 <= reg3_q;
		reg4 <= reg4_q;
		reg5 <= reg5_q;
		reg6 <= reg6_q;
		reg7 <= reg7_q;
	end Behavioral;
```

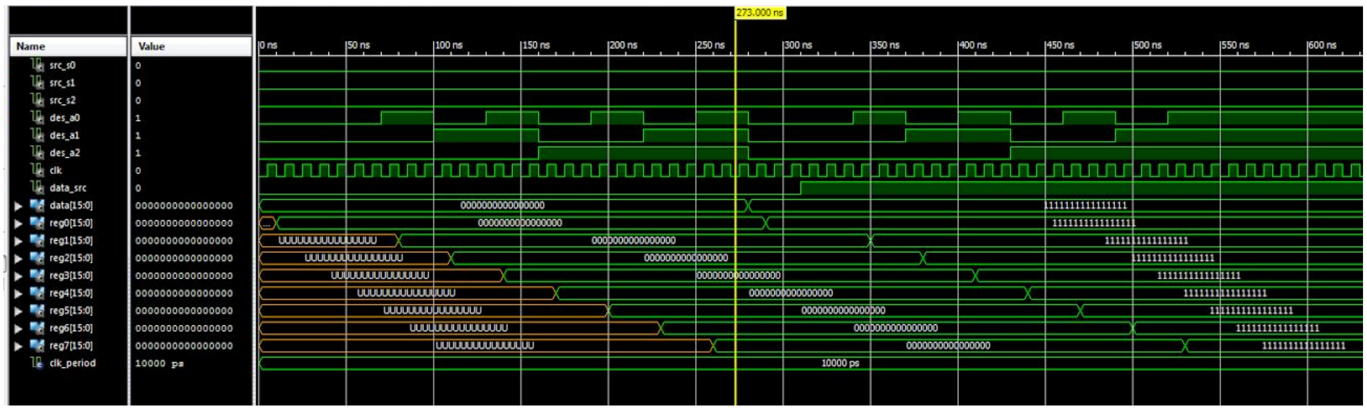
2.2 Register_file Test Bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY register_file_tb IS
END register_file_tb;

ARCHITECTURE behavior OF register_file_tb IS

        -- Component Declaration for the Unit Under Test (UUT)
        COMPONENT register_file
        PORT(
                src_s0, src_s1, src_s2: IN  std_logic;
                des_A0, des_A1, des_A2: IN  std_logic;
                CLK: IN  std_logic;
                data_src: IN  std_logic;
                data: IN  std_logic_vector(15 downto 0);
                reg0, reg1, reg2, reg3, reg4, reg5, reg6, reg7: out  STD_LOGIC_VECTOR (15
             downto 0));
        END COMPONENT;


      --Inputs
      signal src_s0: std_logic:= '0';
      signal src_s1: std_logic:= '0';
      signal src_s2: std_logic:= '0';
      signal des_A0: std_logic:= '0';
      signal des_A1: std_logic:= '0';
      signal des_A2: std_logic:= '0';
      signal CLK: std_logic:= '0';
      signal data_src: std_logic:= '0';
      signal data: std_logic_vector(15 downto 0):= (others => '0');

      --Outputs
      signal reg0, reg1, reg2, reg3, reg4, reg5, reg6, reg7: STD_LOGIC_VECTOR (15 downto 0);

      -- Clock period definitions
      constant CLK_period: time := 10 ns;

BEGIN

      -- Instantiate the Unit Under Test (UUT)
      uut: register_file PORT MAP (
              src_s0 => src_s0,
              src_s1 => src_s1,
              src_s2 => src_s2,
              des_A0 => des_A0,
              des_A1 => des_A1,
              des_A2 => des_A2,
              CLK => CLK,
              data_src => data_src,
              data => data,
              reg0 => reg0,
                        reg1 => reg1,
                        reg2 => reg2,
                        reg3 => reg3,
                        reg4 => reg4,
                        reg5 => reg5,
                        reg6 => reg6,
                        reg7 => reg7);

            -- Clock process definitions
            CLK_process :process
            begin
                        CLK <= '0';
                        wait for CLK_period/2;
                        CLK <= '1';
                        wait for CLK_period/2;
            end process;


    -- Stimulus process
    stim_proc: process
    begin
                        -- hold reset state for 100 ns.
                        wait for 10 ns;

                        -- Q2(a) Loading straight into the registers
                        src_S0<='0'; src_S1<='0'; src_S2<='0';
                        data_src <= '0';
                        data <= "0000000000000000";

                        -- 0
                        wait for CLK_period*3;
                        des_A0<='0'; des_A1<='0'; des_A2<='0';

                        -- 1
                        wait for CLK_period*3;
                        des_A0<='1'; des_A1<='0'; des_A2<='0';

                        -- 2
                        wait for CLK_period*3;
                        des_A0<='0'; des_A1<='1'; des_A2<='0';

                        -- 3
                        wait for CLK_period*3;
                        des_A0<='1'; des_A1<='1'; des_A2<='0';

                        -- 4
                        wait for CLK_period*3;
                        des_A0<='0'; des_A1<='0'; des_A2<='1';

                        -- 5
                        wait for CLK_period*3;
                        des_A0<='1'; des_A1<='0'; des_A2<='1';

                        -- 6
```

```
		wait for CLK_period*3;
		des_A0<='0'; des_A1<='1'; des_A2<='1';

		-- 7
		wait for CLK_period*3;
		des_A0<='1'; des_A1<='1'; des_A2<='1';

		-- Q2(b) Loading from one register into the others
		wait for CLK_period*3;

		-- setup to read value from 0
		des_A0<='0'; des_A1<='0'; des_A2<='0';
		data <= "1111111111111111";
		wait for CLK_period*3;
		data_src <= '1';

		-- 1
		wait for CLK_period*3;
		des_A0<='1'; des_A1<='0'; des_A2<='0';

		-- 2
		wait for CLK_period*3;
		des_A0<='0'; des_A1<='1'; des_A2<='0';

		-- 3
		wait for CLK_period*3;
		des_A0<='1'; des_A1<='1'; des_A2<='0';

		-- 4
		wait for CLK_period*3;
		des_A0<='0'; des_A1<='0'; des_A2<='1';

		-- 5
		wait for CLK_period*3;
		des_A0<='1'; des_A1<='0'; des_A2<='1';

		-- 6
		wait for CLK_period*3;
		des_A0<='0'; des_A1<='1'; des_A2<='1';

		-- 7
		wait for CLK_period*3;
		des_A0<='1'; des_A1<='1'; des_A2<='1';

		wait;
	end process;
END;
```

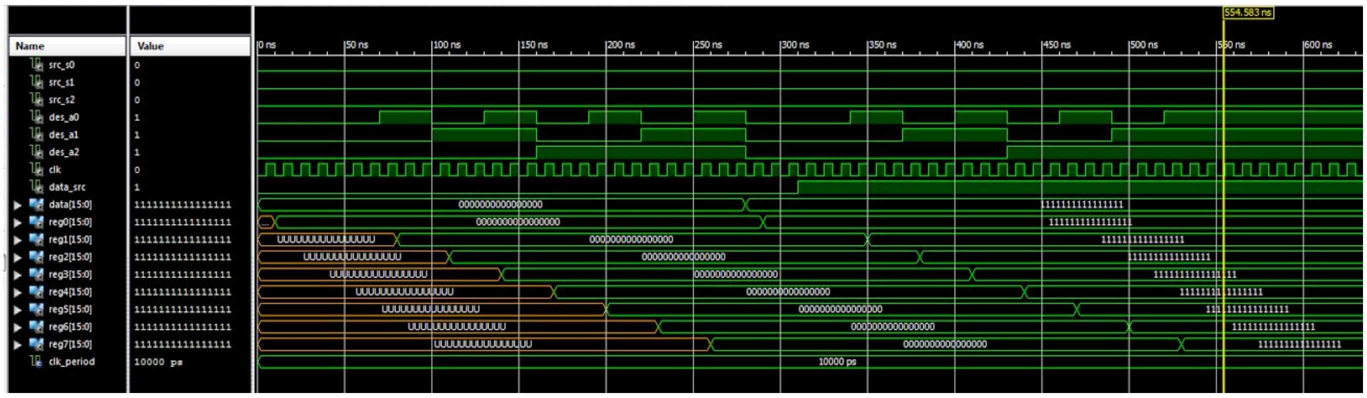
2.2 Register_file Test Results

Loading raw data “0000000000000000” into the registers:



Name	Value
src_s0	0
src_s1	0
src_s2	0
des_a0	1
des_a1	1
des_a2	1
clk	0
data_src	0
data[15:0]	0000000000000000
reg0[15:0]	0000000000000000
reg1[15:0]	0000000000000000
reg2[15:0]	0000000000000000
reg3[15:0]	0000000000000000
reg4[15:0]	0000000000000000
reg5[15:0]	0000000000000000
reg6[15:0]	0000000000000000
reg7[15:0]	0000000000000000
clk_period	10000 ps

Loading saved data “1111111111111111” from reg0 into the other registers:



Name	Value
src_s0	0
src_s1	0
src_s2	0
des_a0	1
des_a1	1
des_a2	1
clk	0
data_src	1
data[15:0]	1111111111111111
reg0[15:0]	1111111111111111
reg1[15:0]	1111111111111111
reg2[15:0]	1111111111111111
reg3[15:0]	1111111111111111
reg4[15:0]	1111111111111111
reg5[15:0]	1111111111111111
reg6[15:0]	1111111111111111
reg7[15:0]	1111111111111111
clk_period	10000 ps